**深 圳 大 学 实 验 报 告**

**课程名称： 计算机系统(2)**

**实验项目名称： Cache实验**

**学院： 计算机与软件学院**

**专业： 计算机与软件学院所有专业**

**指导教师： 刘 刚**

**报告人：何泽锋 学号：2022150221班级： 高性能特色班**

**实验时间： 2024年 6 月 7 日至 6 月 10 日**

**实验报告提交时间： 2024年 6月 21 日**

**教务处制**

|  |
| --- |
| **一、实验目的：**   1. 加强对Cache工作原理的理解； 2. 体验程序中访存模式变化是如何影响cahce效率进而影响程序性能的过程； 3. 学习在X86真实机器上通过调整程序访存模式来探测多级cache结构以及TLB的大小。 |
| **二、实验环境**  X86真实机器 |
| **三、实验内容和步骤**  **1、分析Cache访存模式对系统性能的影响**   * 1. 给出一个矩阵乘法的普通代码A，设法优化该代码，从而提高性能。   2. 改变矩阵大小，记录相关数据，并分析原因。   **2、编写代码来测量x86机器上（非虚拟机）的Cache 层次结构和容量**   1. 设计一个方案，用于测量x86机器上的Cache层次结构，并设计出相应的代码； 2. 运行你的代码获得相应的测试数据； 3. 根据测试数据来详细分析你所用的x86机器有**几级Cache**，**各自容量**是多大？ 4. 根据测试数据来详细分析**L1 Cache行**有多少？   **4、尝试测量你的x86机器TLB有多大？（选做）**  代码A：  #include <sys/time.h>  #include <unistd.h>  #include <stdlib.h>  #include <stdio.h>  int main(int argc, char \*argv[])  {  float \*a,\*b,\*c, temp;  long int i, j, k, size, m;  struct timeval time1,time2;    if(argc<2) {  printf("\n\tUsage:%s <Row of square matrix>\n",argv[0]);  exit(-1);  } //if  size = atoi(argv[1]);  m = size\*size;  a = (float\*)malloc(sizeof(float)\*m);  b = (float\*)malloc(sizeof(float)\*m);  c = (float\*)malloc(sizeof(float)\*m);  for(i=0;i<size;i++) {  for(j=0;j<size;j++) {  a[i\*size+j] = (float)(rand()%1000/100.0);  b[i\*size+j] = (float)(rand()%1000/100.0);  }  }    gettimeofday(&time1,NULL);  for(i=0;i<size;i++) {  for(j=0;j<size;j++) {  c[i\*size+j] = 0;  for (k=0;k<size;k++)  c[i\*size+j] += a[i\*size+k]\*b[k\*size+j];  }  }  gettimeofday(&time2,NULL);    time2.tv\_sec-=time1.tv\_sec;  time2.tv\_usec-=time1.tv\_usec;  if (time2.tv\_usec<0L) {  time2.tv\_usec+=1000000L;  time2.tv\_sec-=1;  }    printf("Executiontime=%ld.%06ld seconds\n",time2.tv\_sec,time2.tv\_usec);  return(0);  }//main |
| **四、实验结果及分析**  **1、分析Cache访存模式对系统性能的影响**  表1、普通矩阵乘法与及优化后矩阵乘法之间的性能对比   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 矩阵大小 | 100 | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | | 一般算法执行时间 | 0.003251 | 0.350898 | 2.884303 | 12.926547 | 27.282556 | 61.446995 | 99.092194 | | 优化算法执行时间 | 0.00278 | 0.300608 | 2.378763 | 7.92293 | 18.660799 | 37.290718 | 69.272728 | | 加速比  speedup | 1.169424 | 1.167294 | 1.212522 | 1.6315361 | 1.4620251 | 1.6477826 | 1.4304647 |   加速比定义：加速比=优化前系统耗时/优化后系统耗时；  所谓加速比，就是优化前的耗时与优化后耗时的比值。加速比越高，表明优化效果越明显。  分析原因：代码的作用是进行矩阵的乘法，一般算法中矩阵乘法的是按照原理进行相乘，及b矩阵是按照列优先进行计算，此时cache缓存的数据命中率很低，需要频繁访问内存，造成获取数据的时间较长，程序运行时间长。因此在存储和访问b时都将行和列倒转，此时即可按照行优先进行访问，充分利用cache缓存相邻数据的优势，减少对内存的访问，进而程序运行效率更高。  **2、测量分析出Cache 的层次结构、容量以及L1 Cache行有多少？**   1. 实验原理；   Cache一般分为三层\* 0，分别是L1、L2、L3，每一层之间的读取速率不同，但访问的数据不在当前层时，系统会往更高的层寻找。因此通过控制数据的范围，来确定访问的范围，进而确定每一层Cache能缓存的数据大小   1. 测量方案及代码；   分别测量Cache 的层次结构、容量和L1 Cache行大小  1）测量Cache 的层次结构、容量  ①方案：逐渐增大传入的比特数，按照传入的比特数，创建对应大小的字符组，产生大量需要访问的位置，每个位置都在比特数范围内，将访问的位置随机化存储，及遍历时不是按照字符组顺序遍历，而是在范围内随机遍历，这样可以判断得到cache缓存的字节数边界，但数据存储范围大于cache时，多次随机遍历可以增加cache重新读取的次数，进而产生时间差  ②代码：    2）测量L1 Cache行大小  ①方案：基本逻辑同上，传入的数据变为每次读取的比特数，保证每次数据总个数相同，为了确保不同长度下都读取一样次数，使用两层循环，内层循环每次按照读取长度遍历，外层循环再遍历长度次，这样两层相乘结果可以保证读取次数相同，代码时间差异将会由cache缓存的大小决定  ②代码：     1. 测试结果；   ①Cache 的层次结构、容量测试结果      ②L1 Cache行大小测试结果       1. 分析过程；   ①根据表格数据可以知道，在1536(1.5MB)-3072(3MB)范围内出现一次变化，判断此范围是L1的临界范围，进一步观察可以发现在32768(32MB)-34816(34MB)范围内存在第二次较大变化，因此判断为L2的临界。最后在34816(34MB)-38912(38MB)范围内是第三次变化，所以L3应该在此范围  ②根据表格数据并结合图表变化可以推测，但长度大于16B时增长较快，因此推断cache line的大小为16B   1. 验证实验结果。   查看任务管理器中的进程可以知道，cache存在3层结构（L1、L2、L3），缓存的大小分别为2.1MB、32MB、36MB，与实验得到的范围接近，因此结果正确    **3、尝试测量你的x86机器TLB有多大？（选做）**  （1）测量原理  假设有一个横跨多内存页的大数组a，页面大小是4096个字节，int大小为4个字节，那么每页就有1024个数组元素。依次往后访问页面即可测量出TLB容量大小  开一个横跨多页的数组，依次修改各个页里面的数组元素，访问的页面数量逐渐变多，直到超过了TLB的容量，访问数组元素就会变慢。因此通过时间变化即可确定TLB容量即TLB大小  （2）代码    （3）结果分析，可以看到TLB的缓存也是分层的，此处第一个波峰的数据为page=4，即4个条目，已知内存页的大小为4KB，因此此处测量结果TLB的大小为4\*4KB，即16KB |
| **五、实验结论**  通过本次实验，基本了解了如何测试电脑的cache。实验结果显示，该主机cache含有三个层级，L1、L2、L3。经过测量可知L1大小约为2MB，L2大小约为32MB，L3大小约为36MB。任务二测量cache行的大小，通过依次增大传入数据长度可以测得cache line的大小为16B。任务三测量TLB大小，查阅资料可知，TLB是用与缓存虚拟地址和物理地址的映射，因此通过扩大虚拟地址的范围，可以测得缓存的地址范围，实验测得TLB也含有多层，实验结果TLB第一层为16KB。要注意的是，在测量TLB的容量时，代码在cpu的单核上运行，这样可以减少多核之间的互相影响，使得结果更加准确 |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2024年 月 日 |
| 备注： |